LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-636185

A Python Object Oriented & Data Driven Testing Infrastructure for an Environmental Information Management System

E. Barbosa, G. W. Laguna

May 7, 2013

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

A Python Object Oriented & Data Driven Testing Infrastructure for an Environmental Information Management System

E. Barbosa, *Inter American University of Puerto Rico*
G. Laguna, mentor, *Lawrence Livermore National Laboratory*

Abstract

Software systems such as TEIMS, the Taurus Environmental Information Management System evolve in size and complexity making manual testing no longer sufficient or feasible, becoming a performance and quality risk. TEIMS is a system of systems comprised of dynamic, data-driven web-based applications that support planning, operational monitoring, data collection, sample management, reporting and, scientific findings for LLNL's Environmental Restoration Department (ERD). Selenium's Integrated Development Environment (IDE) was previously used to support TEIMS testing. An approach with limited flexibility for future software development. As a student intern I was assigned the task to develop an object oriented testing infrastructure for TEIMS. A critical resource in this approach is the development of a catalogue of page elements. To accomplish this I created a comma separated value file, which is comprised of key identifiers based on page elements. Using the Python language and the Selenium Web driver I then developed the object mapping structure, which serves as a wrapper for exposure of each element allowing interaction with the tests and test suites. A master driver "Module" allowed the web driver to be shared among the objects and tests. I then developed a web-based graphical user interface (GUI) for test history, metrics and searching. This data driven approach provides flexibility and extensibility, allowing changes to the structure to be performed in a single place. The result is an object oriented/data driven infrastructure that allows tests to be simpler, flexible, and maintainable, with searchable results.

Introduction

The development of tests during software engineering is an important component of development for production quality software. Testing for TIEMS was previously done using the Selenium IDE. Selenium IDE is a testing Framework, based on the Selenese Language, which provides user interaction recording capabilities and replay among other features. Barbosa, E. (2012). Report on Selenium IDE. This testing strategy provides coverage with limited flexibility to change for software gradually updating over time or new development.

The Python Testing Infrastructure

Extensive systems such as TEIMS can benefit from an object oriented & data driven testing infrastructure. The Python testing infrastructure is a system of systems such as TEIMS. The testing infrastructure is designed to support-testing capabilities built in compliance with each software's structural needs. Such structure will make testing capabilities easily extensible to support future software upgrades and new development. The first step was the creation of a catalogue file comprised of key identifying elements obtained from the web applications to be tested. The catalogue was built as a CSV, Comma Separated Value file.

This catalogue will allow a data driven approach for the creation of an Object Map structure. Second, an Object Map Module was designed. This structure is built from a series of methods and functions that are driven by the catalogue. This Module provides a powerful testing environment for all elements located in the web application that is being tested. The creation of such testable objects allows element and data collection

manipulation providing a wrapping structure for each software. To build an Object Map Module this should dynamically initialize the Master Driver and the url which the Module is being built for. Next is the creation of a CSV reader for the catalogue, the reader will iterate over each element and parse the data needed to create a dictionary for the web application. The dictionaries are used by the units in the Object Map Module.

Third, a Master Driver Module was created using the Selenium Firefox Web-driver. Selenium's Web-driver extends browsers such as; Firefox, Chrome and IE. The Selenium API, Application Programming Interface allows programs or scripts to manipulate the browser. The development of the Master Driver provides multiple usages through the testing infrastructure; the test cases, test suites and the object map structure. The Master Driver Module will be shared from the Test Case Scripts and Test Suites to each Object Map being used during execution of testing, see Figure 1.

Object Oriented Testing

Test Scripts, Suites and Object Map are developed implementing an object oriented Unit approach. Unit testing is the process which individual units of source code, sets of one or more program modules together with associated control data, usage procedures, and operating procedures, are tested. The developments of each Test Script and Suites have been designed with a fixture that manipulates a build and break behavior of each unit. During the build process two steps must be followed, first the Object Map Module and Master Driver are imported to the Test Script and combined to drive the testing environment. Providing the Master Driver to be shared between both Modules

Object Map and Test Script at a single time. Such dependency allows the Test Scripts to manipulate elements from the Object Map Module.

Second, the build will create an object of the CSV catalogue. The instance of the CSV reader will prepare the Object Map with the proper dependencies for it's relation with the web application. Between the build and break down process the Test Case is designed to fetch the targeted web application address located in the Object Map Module and begin the testing. For the process of building Test Cases such actions can be performed; Test Scripts can request, compare and send data to and receive from the units in the Object Map. During the break down this will create an object with an instance of a close method, from the Object Map Module. The object will close the catalogue and quit the Master Driver. Now all Tests Scripts can be executed when desired from terminal, providing a unit-test detailed shell based error log.

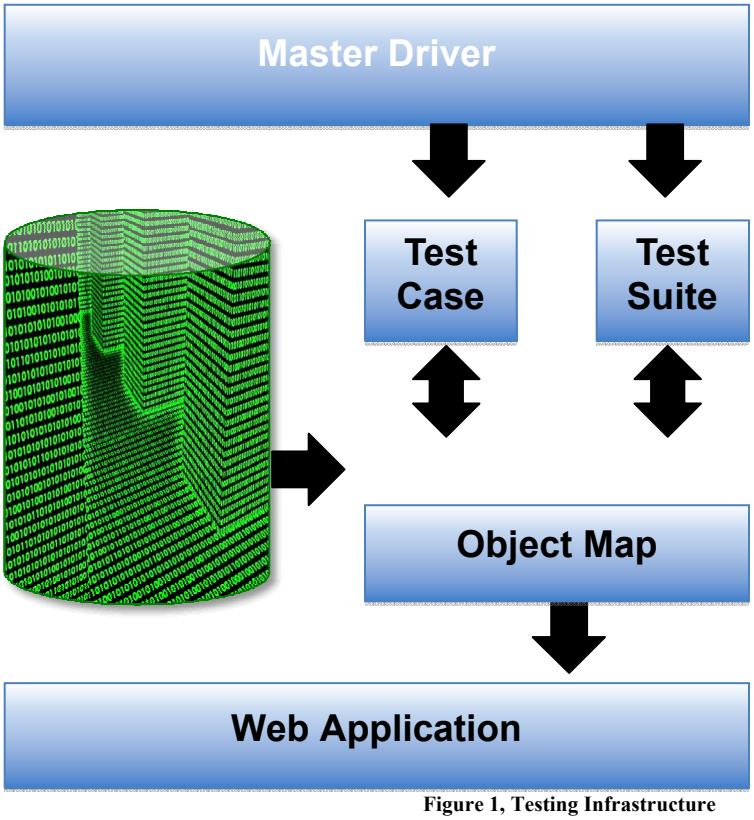

Figure 1, Testing Infrastructure

Terminal Test Manager

As an addition to running test scripts from terminal and viewing an output result in the shell. I developed a terminal based Test Manager. To create the Test Manager I first developed a Runner Module. The Runner was developed using an HTML test runner. The purpose of the Runner Module is to generate a detailed report of each Test or Suite run. A Rollup Script was integrated in each Test and Suite, which provides a portability of execution. The Test Manager will integrate both Rollup Script and Runner, query all Test Script options and output to the user. Allowing the user to select which tests will be run and as a result generate a detailed test report, see Figure 2.

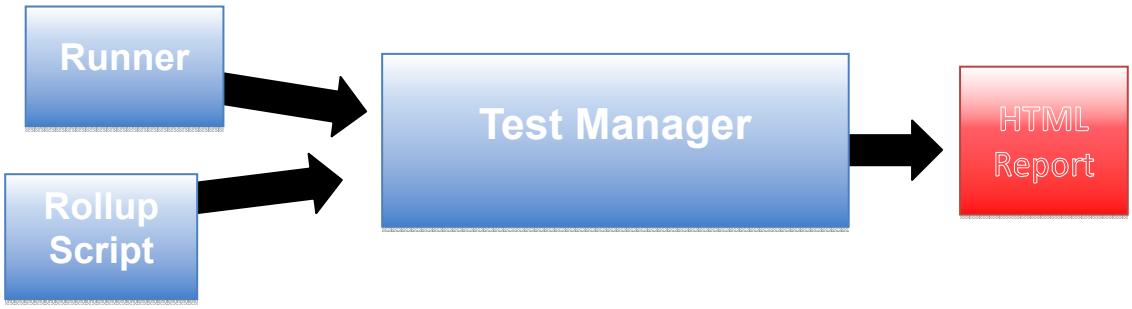

Figure 2, Terminal Based Test Manager

Test Manager GUI

The Test Manager GUI, Graphical User Interface was designed to provide users manageability of detailed testing reports. The GUI was developed by implementing the model view controller paradigm. Barbosa (2012) reports that Model View Controller is a design pattern used in service architectures, MVC separates a software architecture into three distinct elements: model, view and, controller. I first developed a Perl, HTML and

JavaScript a web based Test Manager GUI, Graphical User Interface. The GUI allow for detailed test reports to be queried and displayed for testing metrics capabilities, see Figure 3.

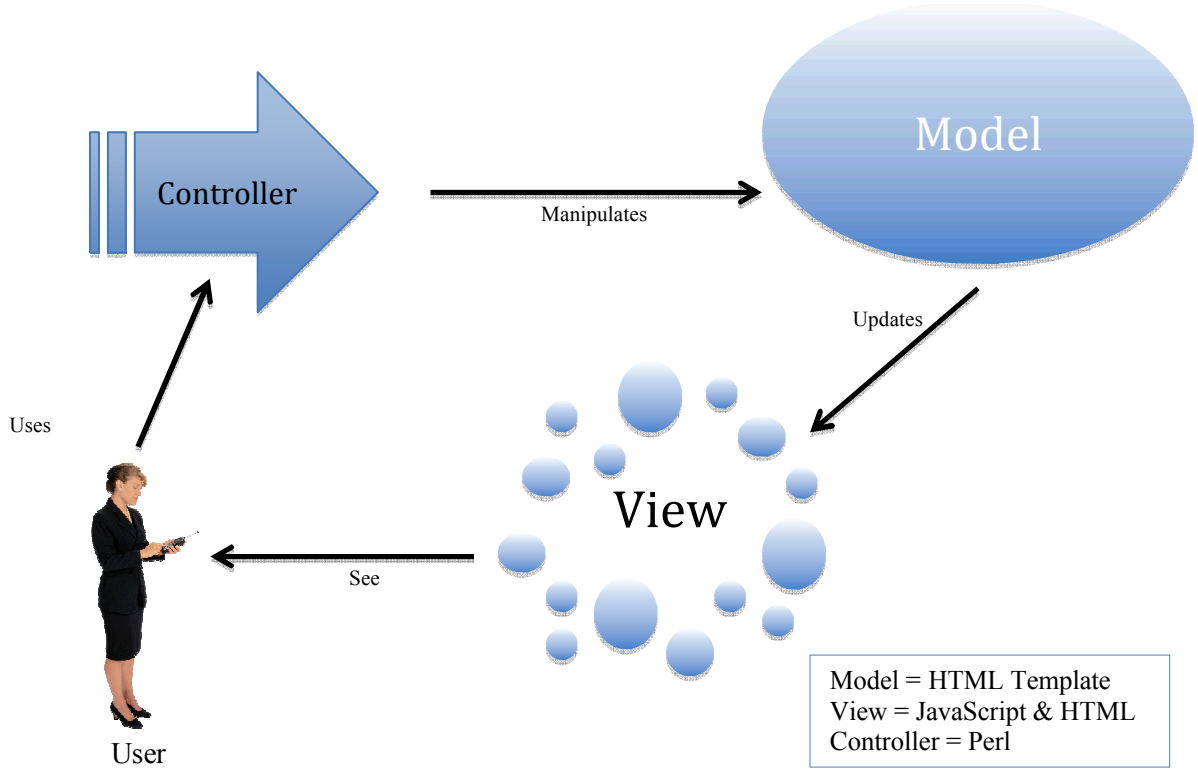

Figure 3, Test Manager GUI

Conclusion

A collection of catalogue-driven methods and functions for manipulating web data and elements. Powerful, wrapped, Object Map Module for each software structure provides flexibility and extensibility, allowing changes to the structure to be made in a single place. An object oriented/data driven testing infrastructure that extends the task of testing to be simple, flexible, and maintainable with searchable results.

Acknowledgments

I am sincerely thankful for my technical mentors guidance when needed, Gary Laguna. The Institute of Scientific Computing, ISCR for supporting my visit and especially the organization, Lawrence Livermore National Laboratory. Last but not least, thank you DOE, Department of Energy, for providing the funds needed to make this knowledge enriching experience possible.

Quotes

Barbosa, E. (2012). *Re-architecting TEIMS Web-based Legacy Applications Using the Model View Controller Paradigm*

References

Unmesh, G. (2012). *Selenium Testing Tools Cookbook.* Burmingham: Packt Publishing

Barbosa, E. (2012). *Web Based Testing for an Environmental Information Management System*